Dual Fixed-Size Ordinally Forgetting Encoding (FOFE) for Competitive Neural Language Models

Sedtawut Watcharawittayakul[*] and **Mingbin Xu**[*] and **Hui Jiang**
Department of Electrical Engineering and Computer Science
Lassonde School of Engineering, York University
4700 Keele Street, Toronto, Ontario, Canada
{watchara, xmb, hj}@eecs.yorku.ca

Abstract

In this paper, we propose a new approach to employ the fixed-size ordinally-forgetting encoding (FOFE) (Zhang et al., 2015b) in neural languages modelling, called *dual-FOFE*. The main idea behind dual-FOFE is that it allows to use two different forgetting factors so that it can avoid the trade-off in choosing either small or large values for the single forgetting factor in the original FOFE. In our experiments, we have compared the dual-FOFE based neural network language models (NNLM) against the original FOFE counterparts and various traditional NNLMs. Our results on the challenging Google Billion Words corpus show that both FOFE and dual FOFE yield very strong performance while significantly reducing the computational complexity over other NNLMs. Furthermore, the proposed dual-FOFE method further gives over 10% relative improvement in perplexity over the original FOFE model.

1 Introduction

Language modelling is an essential task for many natural language processing (NLP) applications including speech recognition, machine translation and text summarization. The goal of language modelling is to learn the distribution over a sequence of characters or words; this distribution may be utilized for encoding the language structure (e.g. the grammatical structure) as well as extracting information from the corpora (Jozefowicz et al., 2016). In the recent years, the popularity of neural networks (NN) has been a significant driving force for language modelling (LM) research; the well-known NN-LM models includes the feedforward NN-LMs (FNN-LMs) (Bengio et al., 2001, 2003), recurrent NN-LMs (RNN-LMs) (Mikolov et al., 2010; Mikolov and Zweig, 2012) and the long short-term memory (LSTM-LMs) (Hochreiter and Schmidhuber, 1997). Among all, FNN-LMs often have a simpler and more efficient learning process, but they tend to underperform the other NN-LMs due to the limited capability to memorize the long term dependency in natural languages (Zhang et al., 2015b). However this drawback could be addressed by applying the fixed-size ordinally-forgetting encoding (FOFE) to FNN's inputs. FOFE is an encoding method, which relies on the ordinally-forgetting mechanism to encode any word sequence based on the positions of words; this also allows the FOFE code to capture the long-term dependency (Zhang et al., 2015b). As shown in Zhang (2015b), FNN-LMs with FOFE can easily yield comparable performance as other NN-LMs. The key parameter in the FOFE method is the forgetting factor, which is responsible for determining the degree of sensitivity of the encoding with respect to the past context. However, the choice of a good value for the forgetting factor could be tricky since both small and large forgetting factors are offering different benefits.

In this paper, we propose a simple alteration to FOFE method, which allows to incorporate two forgetting factors into the fixed-size encoding of the variable-length word sequences. We name this approach as *dual-FOFE*. Our hypothesis is that by incorporating both the small and large forgetting factors in the FOFE encoding, the dual-FOFE is able to simultaneously optimize the abilities to capture the positional information as well as to model long term dependency. In our experiments, we have evaluated the proposed dual FOFE models on two large scale language modeling tasks, namely *enwiki9* and *Google Billion Words (GBW)* corpora. Experimental results have shown that both FOFE models yield very competitive performance on these tasks, comparable with the state-

[*]Equal contribution.

of-the-art systems but with significantly reduced learning complexity. Furthermore, the proposed dual-FOFE method further gives over 10% relative improvement in perplexity over the original FOFE model.

2 Related Work

In this section, we will briefly review the NN-LMs and the original FOFE method. The general idea behind NN-LM is to project the discrete words onto a continuous space, then learn to estimate the conditional probabilities of each known word within the projected space. The training of NN-LMs are often incredibly slow due to the inefficiency of softmax normalization when applied to the extremely large output layer. The solution currently used by many NN-LMs (including our models in this work) is to use noise contrastive estimation (NCE) (Gutmann and Hyvrinen, 2010). The basic idea of NCE is to reduce the probability estimation problem into a probabilistic binary classification problem (Mnih and Teh, 2012; Mnih and Kavukcuoglu, 2013).

2.1 Fixed-Size Ordinally Forgetting Encoding

Fixed-size ordinally-forgetting Encoding (FOFE) is an encoding method which generates a fixed-size representation, namely the FOFE code, for any variable-length word sequence (Zhang et al., 2015b). For a given word sequence $S = \{w_1, w_2, ..., w_T\}$, let e_t denote the one-hot representation of the word w_t, z_t for the FOFE code of the partial word sequence up to word w_t, z_t is computed as follows:

$$z_t = \alpha \cdot z_{t-1} + e_t \quad (1 \le t \le T) \tag{1}$$

where α ($0 < \alpha < 1$) denotes the forgetting factor, a parameter responsible for determining the degree of influence each time step of the past context has on the FOFE code. Obviously, FOFE can convert any variable-length sequence into a fixed-size code with length equal to the size of vocabulary.

In regard to uniqueness of FOFE code, the code is said to be (almost) unique under the two theorems (proven in Zhang (2015a)):

Theorem 1 *If $0 < \alpha \le 0.5$, then FOFE code is guarantee uniqueness for any values of vocabulary's size and sequence's length.*

Theorem 2 *If $0.5 < \alpha < 1$, then FOFE code is guarantee almost uniqueness for any finite values of vocabulary's size and sequence's length, except for a finite set of countable choices of α.*

Furthermore, the chance of actually having any collisions for α between 0.5 and 1 is nearly impossible in practice, due to quantization errors in real computer systems. Hence in practice, it is safe to argue that FOFE is able to uniquely encodes any variable-length sequence into a fixed-size representation.

2.2 FOFE for FNN-LMs

The idea of FOFE based NN-LMs is to use FOFE to encode the partial history sequence of past words in a sentence, then feed this fixed-size FOFE code to a feedforward neural network as an input to predict next word. As shown in Figure 1, the FOFE code could be efficiently computed via time-delayed recursive structure, where the symbol z^{-1} in the figure represents a unit time delay (or equivalently a memory unit) from z_t to z_{t-1}.

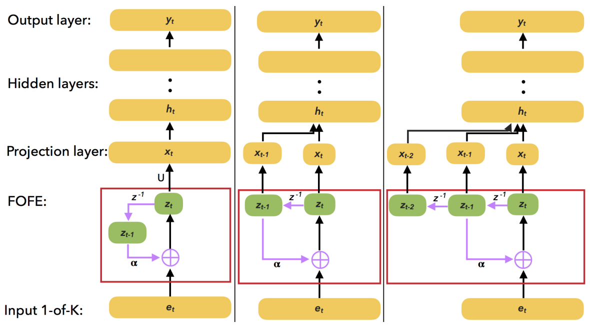

Figure 1: (Left) 1st-order FOFE FNN-LM; (Center) 2nd-order FOFE FNN-LM; (Right) 3rd-order FOFE FNN-LM.

The basic architecture for FOFE based FNN-LMs (called 1st-order) is the standard FNN architecture with an additional layer for encoding the input into FOFE code. However, in this work, we use the 2nd-order and 3rd-order FOFE FNN-LMs, which are shown to produce slightly better results (Zhang et al., 2015b). In a 2nd-order FOFE model, both the current partial sequences FOFE code (denoted as z_t) and the previous partial sequences FOFE code (denoted as z_{t-1}) are utilized to predict next word. In a 3rd-order FOFE model, all z_t, z_{t-1} and z_{t-2} are used as inputs to neural networks.

More recently, the FOFE methods have been successfully applied to many NLP tasks, including word embedding (Sanu et al., 2017), named entity recognition (Xu et al., 2017a), entity discovery and linking (Xu et al., 2016, 2017b).

3 Dual-FOFE

The main idea of dual-FOFE is to generate augmented FOFE encoding codes by concatenating two FOFE codes using two different forgetting factors. Each of these FOFE codes is still computed in the same way as the mathematical formulation shown in Equation (1). The difference between them is that we may select to use two different values for the forgetting factor (denoted as α) for additional modeling benefits.

3.1 Intuition behind Dual-FOFE

As mentioned in the subsection 2.1, the values in a FOFE code are used to encode both the content and the order information in a sequence. This is achieved by a recursive encoding method where at each recursive step the code will be multiplied by the forgetting factor (α) whose value is bounded by $0 < \alpha < 1$. In a practical computer with finite precision, this has an impact on the FOFE's abilities to precisely memorize the long-term dependency of past context as well as to properly represent the positional information.

The FOFE's ability to represent the positional information would improve with smaller forgetting factors. The reason is that that when α is small, the FOFE code (z_t) for each word vastly differs from its neighbour in magnitude. If α is too large (close to 1), the contribution of a word may not change too much no matter where it is. This may hamper the following neural networks to model the positional information. Conversely, the FOFE's ability to model the long-term dependency of the older context would improve with larger forgetting factors. This is because when α is small, the contribution of a word from the older history may quickly underflow to become irrelevant (i.e. forgotten) when computing the current word.

In the original FOFE with just a single forgetting factor, we would have to determine the best trade-off between these two benefits. On the other hand, the dual-FOFE does not face such issues since it is composed of two FOFE codes: the half of the dual-FOFE code using a smaller forgetting factor is solely optimized and responsible for representing the positional information of all words in the sequence; meanwhile the other half of the dual-FOFE code using a larger forgetting factor is optimized and responsible for maintaining the long-term dependency of past context.

3.2 Dual-FOFE based FNN-LM

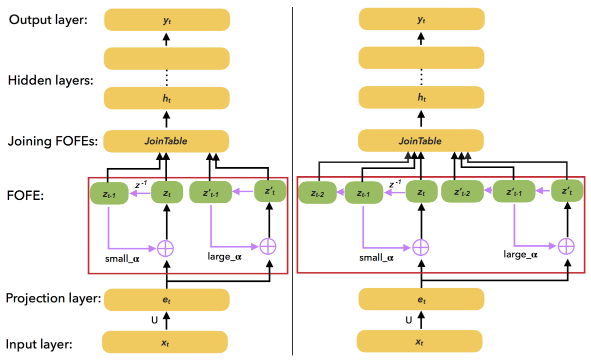

Figure 2: (Left) 2nd-order Dual-FOFE FNN-LM; (Right) 3rd-order Dual-FOFE FNN-LM.

As shown in Figure 2, the architecture of dual-FOFE based FNN-LMs is very similar to the original FOFE FNN-LMs.[1] In the Dual-FOFE FNN-LMs, the input word sequence would have to pass through two branches of the FOFE layers (using two different forgetting factors) and each encoding branch will produce a FOFE code representing the input sequence. These two FOFE codes are then joined to produce the dual-FOFE code, which would be fed to FNNs to predict the next word.

It might also be worth noting that in our implementation we do not explicitly reset FOFE codes, i.e. z_t value, at sentence boundaries. However, far-away histories will be gradually forgotten by the recursive calculation in FOFE due to $0 < \alpha < 1$ and finite precision in computers.

3.3 Dual-FOFE vs. Higher Order FOFE

As mentioned previously in the subsection 2.2, the higer order FOFE codes would utilize both the current and the previous sequence FOFE codes for prediction. Hence similar to dual-FOFE, the higher order FOFE could also maintain the sensitivity to both nearby and faraway context. Obviously a much higher order FOFE code may be required in order to achieve the same effect as dual-FOFE in terms of modelling long-term dependency. In this case, the higher order FOFE may also significantly increase the number of parameters in the input layer. At last, the dual FOFE

[1]The difference in the location of the projection layer between Figure 1 and 2 simply indicates two equivalent ways to do word projection. Figure 1 was originally from Zhang (2015b), but they mentioned in text (without a figure) that it is more efficient to do projection as in Figure 2 and both methods are mathematically equivalent since both projection and FOFE steps are linear.

and the higher order FOFE are largely complementary since we have observed consistent performance gains when combining dual FOFE with either 2nd-order or 3rd-order FOFE in our experiments.

4 Experiments

In this work, we have evaluated the proposed dual-FOFE based FNN-LMs against various traditional neural language models on two corpora: i) *enwik9* corpus: it consists of the first 1 billion bytes of English wikipedia dump, having total size of 170.8 million words; the corpus was divided into three parts: the training set (153M words), the test set (8.9M words), and the validation set (8.9M words); the vocabulary size is limited to 80k words (Zhang et al., 2015b). ii) Google Billion Words (GBW) corpus: it contains about 800 million words and the corpus is divided into two parts: the training set (792M words) and the test set (8M words); the vocabulary size for this corpus is limited to 800k words (Chelba et al., 2013).

4.1 Results on enwiki9

In the experiments on the enwiki9 corpus, we have trained three dual-FOFE FNN-LMs with different forgetting factor pairs, one FOFE FNN-LM, and one tri-FOFE FNN-LM. All five models adopt a 2nd-order FOFE structure, employing a word embeddings of 256 dimensions, three hidden layers of $400, 600, 600$ neurons and an output layer of 80k words (reflecting the vocabulary). [2] Note that the dual-FOFE FNN-LMs have to double the size of input context windows since dual-FOFE essentially contain two FOFE codes. But this increase only accounts for a negligible faction of total model parameters.

As shown in Table 1, all three dual-FOFE FNN-LMs, using three pairs of forgetting factors as (0.5, 0.7) and (0.7, 0.9) and (0.5, 0.9), can significantly outperform other traditional models previously reported on this corpus. We also note that it is beneficial to include a relatively large forgetting factor, such as 0.9, in the dual FOFE models since such a large alpha may help to memorize much longer context in the inputs. When compared with the original FOFE counterpart, the best dual-FOFE model using forgetting factors (0.5, 0.9) offers a relative gain of around 8% in test PPL.

[2]Comparing with Zhang (2015b), our single FOFE FNN-LM baseline use a slightly larger model, which lead to slightly better perplexity.

It is worth noting that our dual-FOFE models can be extended to incorporate more than two alpha values. In fact after we have obtained a strong result supporting our dual-FOFE hypothesis, we have performed additional experiments using three alpha values, the so-called tri-FOFE model. The result on Table 1 has shown that the tri-FOFE FNN-LMs still slightly outperforms the dual-FOFE models. However, the gain is marginal. This leads us to believing that further extension of more alpha values in FOFE would be of limited use.

4.2 Results on Google Billion Words (GBW)

In the experiments on the GBW corpus, we have trained one dual-FOFE FNN-LM and one FOFE FNN-LM. Following the best dual-FOFE model configuration on the previous corpus, this dual-FOFE FNN-LM uses the same pair of dual forgetting factors $(0.5, 0.9)$. Both models adopt a 3rd-order structure, employing word embeddings of 256 dimensions, three hidden layers each of 4096 neurons, a compression layer with 720 neurons, and an output layer of 800k words (reflecting the vocabulary). Although dual-FOFE FNN-LM has doubled the size of input context windows of FOFE FNN-LM, the total number of model parameters in both models are almost equal, roughly 0.82 billion parameters.

As shown in Table 2, the dual-FOFE FNN-LM is able to produce a very competitive performance, comparable with the best previously reported results on this task, such as GCNN-13 (Dauphin et al., 2016) and LSTM-LM (Jozefowicz et al., 2016). The dual-FOFE FNN-LM are among the few single-model systems that are able to achieve test PPL below 40 on this task. Furthermore, our proposed dual FOFE model can significantly reduce the computational complexity, e.g., our model has a relatively smaller number of parameter (0.82B parameters) and it requires much less hardware resource to train (using only 1 GPU in our experiments). When compared with the original FOFE counterpart, the dual-FOFE FNN-LM is able to provide approximately 11% relative improvement in PPL.

5 Conclusions

In this paper, we have proposed a new approach of utilizing the fixed-size ordinally-forgetting encoding (FOFE) method for neural network lan-

Table 1: Test PPL of various LMs on *enwiki9*.

Model	Architecture	PPL
KN 3-gram (Zhang et al., 2015b)	-	156
KN 5-gram (Zhang et al., 2015b)	-	132
FNN-LM 2-gram (Zhang et al., 2015b)	[2*200]-600-600-80k	150
FNN-LM 3-gram (Zhang et al., 2015b)	[3*200]-400-400-80k	131
FNN-LM 4-gram (Zhang et al., 2015b)	[4*200]-400-400-80k	125
RNN-LM (Zhang et al., 2015b)	[1*600]-80k	112
FOFE[α=0.7] FNN-LM (Zhang et al., 2015b)	[2*200]-600-600-80k	107
FOFE[α=0.7] FNN-LM	[2*256]-400-600-600-80k	104.8
Dual-FOFE[α=0.5,0.7] FNN-LM	[2*2*256]-400-600-600-80k	101.7
Dual-FOFE[α=0.7,0.9] FNN-LM	[2*2*256]-400-600-600-80k	97.0
Dual-FOFE[α=0.5,0.9] FNN-LM	[2*2*256]-400-600-600-80k	**96.6**
Tri-FOFE[α=0.5,0.7,0.9] FNN-LM	[3*2*256]-400-600-600-80k	**95.9**

Table 2: Test PPL of various LMs on Google Billion Words.

model	PPL	#param	hardware
Sigmoid-RNN-2048 (Ji et al., 2015)	68.3	4.1B	1 CPU
Interpolated KN 5-gram & 1.1B n-grams (Chelba et al., 2013)	67.6	1.8B	100 CPUs
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33B	-
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3	20B	24 GPUs
LSTM-1024-512 (Jozefowicz et al., 2016)	48.2	0.82B	40 GPUs
LSTM-2048-512 (Jozefowicz et al., 2016)	43.7	0.83B	40 GPUs
LSTM + CNN input (Jozefowicz et al., 2016)	30.0	1.04B	40 GPUs
GCNN-13 (Dauphin et al., 2016)	38.1	-	1 GPU
FOFE[α=0.7] FNN-LM [3*256]-4096*3-720-800k	43.6	0.82B	1 GPU
Dual-FOFE[α=0.5,0.9] FNN-LM [2*3*256]-4096*3-720-800k	**39.0**	**0.82B**	**1 GPU**

guage models (NN-LMs), known as dual-FOFE. As the name implies, this approach involves to produce a new fixed-sized representation for any variable-length sequence from a concatenation of two FOFE codes. This would have allowed us to select two values for the forgetting factors. One FOFE code with a smaller forgetting factor is responsible for representing the positional information of all words in the sequence while the other using a larger forgetting factor is responsible for modelling the even longer term dependency in far away history. Our experiments on both *enwiki9* and Google Billion Words (GBW) tasks have both demonstrated the effectiveness of the dual-FOFE modeling approach. Experimental results on the challenging GBW corpus have shown that the dual-FOFE FNN-LM has achieved over 10% improvement in perplexity over the original FOFE FNN-LM, without any significant drawback in model and learning complexity. When compared with other traditional neural language models, the dual-FOFE FNN-LM has achieved competitive performance with significantly lower computational complexity.

Acknowledgement

This work is supported mainly by a research donation from iFLYTEK Co., Ltd., Hefei, China, and partially by a discovery grant from Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In *Advances in Neural Information Processing Systems 13*, pages 932–938. MIT Press.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3:1137–1155.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083.

Michael Gutmann and Aapo Hyvrinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 297–304. Journal of Machine Learning Research - Proceedings Track.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. 2015. Blackout: Speeding up recurrent neural network language models with very large vocabularies. *CoRR*, abs/1511.06909.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network based language model. In *2012 IEEE Spoken Language Technology Workshop*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2265–2273, USA. Curran Associates Inc.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *ArXiv e-prints*.

Joseph Sanu, Mingbin Xu, Hui Jiang, and Quan Liu. 2017. Word embeddings based on fixed-size ordinally forgetting encoding. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 310–315, Copenhagen, Denmark. Association for Computational Linguistics.

Noam Shazeer, Joris Pelemans, and Ciprian Chelba. 2015. Sparse non-negative matrix language modeling for skip-grams. In *Proceedings of Interspeech 2015*, pages 1428–1432. International Speech Communication Association.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017a. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1237–1247, Vancouver, Canada. Association for Computational Linguistics.

Mingbin Xu, Nargiza Nosirova, Kelvin Jiang, Feng Wei, and Hui Jiang. 2017b. FOFE-based deep neural networks for entity discovery and linking. In *Proceedings of the Text Analysis Conference (TAC) 2017*.

Mingbin Xu, Feng Wei, Sedtawut Watcharawittayakul, Yuchen Kang, and Hui Jiang. 2016. The YorkNRM systems for trilingual EDL tasks at TAC KBP 2016. In *Proceedings of the Text Analysis Conference (TAC) 2016*.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Li-Rong Dai. 2015a. A fixed-size encoding method for variable-length sequences with its application to neural network language models. *CoRR*, abs/1505.01504.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015b. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*, pages 495–500. Association for Computational Linguistics.